Improving Premise Structure in Evolving Takagi-Sugeno Neuro-Fuzzy Classifiers

Abdullah Almaksour, Eric Anquetil

Improving Premise Structure in Evolving Takagi-Sugeno Neuro-Fuzzy Classifiers

Abdullah Almaksour, Eric Anquetil
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
CNRS, UMR IRISA, Campus de Beaulieu, F-35042 Rennes
Université Européenne de Bretagne, France
{*Abdullah.Almaksour, Eric.Anquetil*}*@irisa.fr*

***Abstract*—We present in this paper a new method for the design of evolving neuro-fuzzy classifiers. The presented approach is based on a first-order Takagi-Sugeno neuro-fuzzy model. We propose a modification on the premise structure in this model and we provide the necessary learning formulas, with no problem-dependent parameters. We demonstrate by the experimental results the positive effect of this modification on the overall classification performance.**

***Keywords*-Incremental learning; Takagi-Sugeno; neuro-fuzzy;**

I. Introduction

Classification techniques appear frequently in many application areas, and become the basic tool for almost any pattern recognition task. The main problem in classification is to induce a classifier from a set of data samples. A large amount of samples is needed to set up and evaluate a classification system that can achieve a high accuracy, and it is practically very difficult to have such number of samples covering the whole feature space. Therefore, life-long classifier adaptation becomes more and more an essential point. Moreover, in many application contexts, the classifier needs to take into account new unseen classes and to integrate them in the classification process, which increases the need for "evolving" classifiers. One good example of such applications is the use of online handwriting gesture classifiers which aim at facilitating interactions with computers using pen-based interfaces like whiteboards, tablet PCs, PDA...Etc. The main drawback in the current existing systems is that they are trained "offline" on a specific group of gestures and then implemented to operate without changing their structure during the use. This fixed structure does not allow the user to choose his own set of gestures or to add new ones to assign them to new interactive commands (according to his special needs), for example.

In our work, we aim at building a handwriting classifier, on-the-fly, from scratch and using only few data. Thus, the classifier will be incrementally adapted to achieve high recognition rates as soon as possible and to keep the system robust when introducing new unseen classes at any moment in the life-long learning process.

An incremental learning algorithm is defined in [**?**] as being one that meets the following criteria: it should be able to learn additional information from new data; it should not require access to the original data (i.e. data used to train the existing classifier); it should preserve previously acquired knowledge (it should not suffer from *catastrophic forgetting*, significant loss of original learned knowledge); and it should be able to accommodate new classes that may be introduced with new data. Many of the existing "incremental learning" algorithms are not truly incremental because at least one of the mentioned criteria is violated. These criteria can be briefly expressed by the so called "plasticity-stability dilemma"[**?**], which is that a system must be able to learn in order to adapt to a changing environment but that constant change can lead to an unstable system that can learn new information only by forgetting everything it has so far learned.

We can distinguish two main types of incremental learning algorithms: algorithms for parameter incremental learning and algorithms for structure incremental learning. The incremental learning of parameters can be considered as "adaptation" algorithm. The structure in such systems is fixed and initialized at the beginning of the learning process, and the system parameters are learned incrementally according to newly available data. Some examples of these systems are presented in [**?**] and [**?**].

Most of structure incremental learning algorithms are based on the principle of the ART clustering algorithm [**?**], such as [**?**], [**?**], [**?**]. The main problem

of these systems is that they are sensitive to the selection of the vigilance parameter, to noise level in the training data and to the order in which training data are presented.

In an online incremental learning algorithm, the training set is not available a priori, since the learning examples come over time. Although online learning systems can continuously update and improve their models, not all of them are necessarily based on a real incremental approach. For some systems the model is completely rebuilt at each step of learning using all available data, they are systems with "full instance memory" [**?**]. On the other hand, if the learning algorithm modifies the model using only the last available learning example, it is called a system with "no instance memory" [**?**] [**?**]. A third category is that of systems with "partial instance memory", which select and maintain a reduced subset of learning examples to use them in the next learning step [**?**].

Evolving neuro-fuzzy models [**?**] have been successfully used in many modeling and classification problems during the last years. Several incremental learning algorithms have been proposed for the evolving structure identification and the recursive parameters learning (Denfis[**?**], Flexfis[**?**], eTS[**?**], eTS+[**?**] etc.). One of the most used neuro-fuzzy models is the First-order Takagi-Sugeno fuzzy model. It usually has a linear consequent part and its premise part can be learned from data (no expert is needed). These models can address problems with either single output or multi-output. It consists of a set of fuzzy rules of the following form:

$$\textbf{Rule}_i : \textbf{IF } \vec{x} \textit{ is close to } P_i \textbf{THEN } y_i^1 = l_i^1(\vec{x}), ..., \; y_i^k = l_i^k(\vec{x}) \tag{1}$$

where $l_i^m(\vec{x})$ is the linear consequent function of the rule i for the class m. The Prototype P is defined by a center and a fuzzy zone of influence. A membership function must be defined in order to calculate the "closeness" degree between x and P (considering its center and its fuzzy zone of influence). In the existing approaches, the zone of influence has often a hyper-spherical form [**?**] [**?**]. It has then the same radius for all the feature space dimensions, and it can be represented by a diagonal covariance matrix with identical values on the diagonal. More sophisticated form is presented in [**?**] that allows the zone radius to have different values for the different dimensions (still diagonal covariance matrix but with different values on the diagonal). This ability results in hyper-elliptical zones where the ellipses are still parallel to the feature space axes. In this paper, we go a step ahead in the structure of the fuzzy prototypes in first order TS models by allowing them to have a hyper-elliptical form non-parallel to the feature space axes. This form enables the model to take into consideration the correlations that can exist between the features, and is represented by normal (non-diagonal) covariance matrices. We calculate the prototype centers and covariance matrices in a recursive way with zero problem-dependent parameters. We use a membership function based on a well-known multivariate probability distribution function. By experimenting this structure on our handwritten gesture problem and other benchmark multi-class problems, we show its useful effect on the overall system output and the linear consequent learning. The rest of the paper is organized as follows: We describe in Section **??** the architecture of our neuro-fuzzy classifier. Then, the different elements of the used online incremental learning algorithm are detailed in Section III. We present our experimental results in Section IV before concluding in Section **??**.

II. System architecture

As aforementioned, our system is based on first-order Takagi-Sugeno fuzzy inference system. It consists of a set of fuzzy rules of the following form:

$$\textbf{Rule}_i : \textbf{IF } \vec{x} \textit{ is close to } P_i \textbf{THEN } y_i^1 = l_i^1(\vec{x}), ..., \; y_i^k = l_i^k(\vec{x}) \tag{2}$$

where $l_i^m(\vec{x})$ is the linear consequent function of the rule i for the class m:

$$l_i^m(\vec{x}) = \vec{\pi}_i^m \vec{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \tag{3}$$

where n is the size of the input vector. To find the class of $\vec{x}$, its membership degree $\beta_i(\vec{x})$ to each fuzzy prototype is first computed. After normalizing these membership degrees, the sum-product inference is used to compute the system output for each class:

$$y^m(\vec{x}) = \sum_{i=1}^{r} \bar{\beta}_i(\vec{x}) \; l_i^m(\vec{x}) \tag{4}$$

where r is the number of fuzzy rules in the system. The winning class label is given by finding the maximal output and taking the corresponding class label as response:

$$class(\vec{x}) = y = argmax \; y^m(\vec{x}) \qquad m = 1, .., k \tag{5}$$

The membership degree can be calculated in many ways. For hyper-spherical or axes-parallel hyper-elliptical prototypes, the membership degree can

be computed depending on the prototype center $\vec{\mu_i}$ and the radius value σ_i (the same value in all the dimensions for the former, and different values for the latter). In this case, the Gaussian membership function is generally used. The value of $\beta_i(\vec{x})$ can then be computed as follows:

$$\beta_i(\vec{x}) = \prod_{j=1}^{n} \exp(-\frac{\|x - \mu_i\|_j^2}{2\sigma_{ij}^2}) \tag{6}$$

It must then be normalized as follows:

$$\bar{\beta}_i(\vec{x}) = \frac{\beta_i(\vec{x})}{\sum_{j=1}^{r} \beta_j(\vec{x})} \tag{7}$$

In our model, where the fuzzy influence zones of the prototype take a rotated hyper-elliptical form, the membership degree is computed by the prototype center $\vec{\mu_i}$ and its variance-covariance matrix A_i:

$$A_i = \begin{bmatrix} \sigma_1^2 & c_{12} & ... & c_{1n} \\ c_{21} & \sigma_2^2 & ... & c_{2n} \\ ... & ... & ... & ... \\ c_{n1} & c_{n2} & ... & \sigma_n^2 \end{bmatrix}_i \tag{8}$$

where $c_{12}(= c_{21})$ is the covariance of x_1 and x_2, and so on. In order to calculate the membership degree, we use the multivariate Cauchy probability distribution. The value of $\beta_i(\vec{x})$ is computed in our model as follows:

$$\beta_i(\vec{x}) = \frac{1}{2\pi\sqrt{|A_i|}} \left[1 + (\vec{x} - \vec{\mu_i})^t A_i^{-1} (\vec{x} - \vec{\mu_i})\right]^{-\frac{n+1}{2}} \tag{9}$$

III. ON-LINE INCREMENTAL LEARNING ALGORITHM

The incremental learning algorithm of our model consists of three different tasks: the creation of new rules, the adaptation of the existing rule's premises, and the tuning of the linear consequent parameters. These three tasks must be done in an online incremental mode and all the needed calculation must be completely recursive. We will describe in this section these different parts of the learning algorithm.

A. Incremental clustering

When introducing a new training sample in an online learning mode, it will either reinforce the information contained in the previous data and represented by the current clustering, or bring enough information to form a new cluster or modify an existing one. The importance of a given sample in the clustering process can be evaluated by its *potential* value. The potential of a sample is defined as inverse of the sum of distances between a data sample and all the other data samples [**?**]:

$$Pot_k(\vec{x}(k)) = \frac{1}{1 + \sum_{i=1}^{k-1} \|x(k) - x(i)\|^2} \tag{10}$$

A recursive method for the calculation of the potential of a new sample was introduced in [**?**], which made this technique a promised solution for any incremental clustering problem. The recursive formula avoids memorizing the whole previous data but keeps - using few variables - the density distribution in the feature space based on the previous data:

$$P_k(\vec{x}(k)) = \frac{k-1}{(k-1)\alpha(k) + \gamma(k) - 2\zeta(k) + k - 1} \tag{11}$$

where

$$\alpha(k) = \sum_{j=1}^{n} x_j^2(k) \tag{12}$$

$$\gamma(k) = \gamma(k-1) + \alpha(k-1), \quad \gamma(1) = 0 \tag{13}$$

$$\zeta(k) = \sum_{j=1}^{n} x_j(k)\eta_j(k), \quad \eta_j(k) = \eta_j(k-1) + x_j(k-1), \quad \eta_j(1) = 0 \tag{14}$$

Introducing a new sample affects the potential values of the centers of the existing clusters, which can be recursively updated by:

$$P_k(\mu_i) = \frac{(k-1)P_{k-1}(\mu_i)}{k - 2 + P_{k-1}(\mu_i) + P_{k-1}(\mu_i) \sum_{j=1}^{n} \|\mu_i - x(k-1)\|_j^2} \tag{15}$$

If the potential of the new sample is higher than the potential of the existing centers then this sample will be a center of a new cluster and a new fuzzy rule will be formed in the case of our neuro-fuzzy model. So, the center of the new prototype $\vec{\mu}_{r+1} = \vec{x}_k$ and its covariance matrix $A_{r+1} = \epsilon I$, where I is the identity matrix of size n and ϵ is a problem-independent parameter and can generally be set to 10^{-2}.

B. Premise adaptation

This adaptation process allows to incrementally update the prototype centers coordinates according to each new available learning data, and to recursively compute the prototype covariance matrices in order to give them the rotated hyper-elliptical form. For each new sample $\vec{x}_k$, the center and the covariance matrix of the prototype that has the highest activation degree are updated.

The center coordination of the selected prototype is recalculated as follows:

$$\vec{\mu}_i = (1 - \frac{1}{s_i + 2})\vec{\mu}_i + \frac{1}{s_i + 2}(\vec{x}_k - \vec{\mu}_i) \quad (16)$$

where s_i represents the number of updates that have been already applied on this prototype. The covariance matrix is recursively computed as follows:

$$A_i = (1 - \frac{1}{s_i + 2})A_i + \frac{1}{s_i + 1}(\vec{x}_k - \vec{\mu}_i)(\vec{x}_k - \vec{\mu}_i)^t \quad (17)$$

For practical issues, since the membership degree can be calculated using only $A^{-1}(|A| = \frac{1}{|A^{-1}|})$, and in order to avoid any matrix inversion, we use an updating rule for A^{-1} directly [**?**]:

$$A_i^{-1} = \frac{A_i^{-1}}{1-\alpha} - \frac{\alpha}{1-\alpha} \cdot \frac{(A_i^{-1}(\vec{x}_k - \vec{\mu}_i)) \cdot (A_i^{-1}(\vec{x}_k - \vec{\mu}_i))^t}{1 + \alpha((\vec{x}_k - \vec{\mu}_i)^t A_i^{-1}(\vec{x}_k - \vec{\mu}_i))} \quad (18)$$

where $a = \frac{1}{s_i+1}$.

C. Linear consequent tuning

The tuning of the linear consequent parameters in a first-order TS model can be done by the weighted Recursive Least Square method (wRLS). Let Π_i be the linear consequent parameters of the rule i:

$$\Pi_i = \left[\vec{\pi}_i^1 \;\; \vec{\pi}_i^2 \;\; ... \;\; \vec{\pi}_i^m\right]^t \quad (19)$$

where $\vec{\pi}_i^c = [a_{i0}^c \;\; a_{i1}^c \;\; ...a_{in}^c]$. It can be recursively estimated as follows:

$$\Pi_i = \Pi_i + C_i\bar{\beta}_i(\vec{x}_k)\vec{x}_k(Y_k - \vec{x}_k\Pi_i), \;\; \Pi_{init} = \vec{0} \quad (20)$$

$$C_i = C_i - \frac{\bar{\beta}_i(\vec{x}_k)C_i\vec{x}_k\vec{x}_k^t C_i}{1 + \bar{\beta}_i(\vec{x}_k)\vec{x}_k^t C_i\vec{x}_k}, \;\; C_{init} = \Omega I \quad (21)$$

where Ω is a large positive number, and I is the identity matrix.

IV. Experimental results

A. Handwritten gesture recognition

We will particularly focus in our experiments on the rapidity of the performance improvement in the beginning of the incremental learning process, and on the stability and the recovery speed of the performance when introducing new unseen classes. We led the experiments on the "SIGN" database, which is a database of on-line handwritten gestures (figure **??**). It is composed of 25 different gestures drawn by 17 different writers on Tablet PCs. Each writer has drawn 100 samples of each gesture, *i.e.* 1,700 gestures in each writer-specific dataset. The dataset (and additional information on the data collection protocol) can be found in [**?**]. Each gesture is described by a set of 10 features. The presented

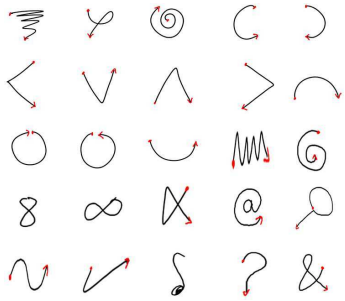


Figure 1: The used handwritten gestures

results are the average of results of 17 different tests for the 17 writers. In order to get the results unbiased by the data order effect, we repeat the experiment for each writer 40 times with different random data orders and the mean results are considered. We used about half of the database for the incremental learning process and the rest is used to estimate the evolution of the performance during the learning process. Two fuzzy incremental learning models are compared in these experiments:

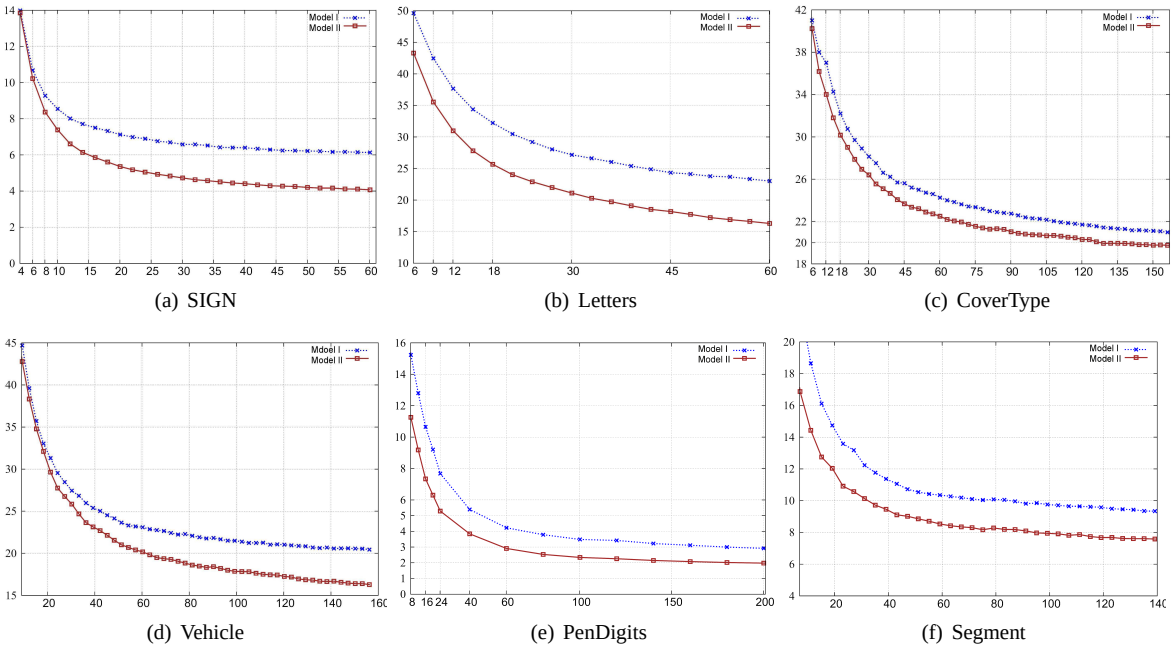
- Model I: evolving first-order TS classifier with parallel-to-axes hyper-elliptical prototypes,
- Model II: our extended version with rotated hyper-elliptical prototypes (covariance between features are considered).

We introduce the 25 gestures in the beginning of the learning process. Two new samples from each class are presented to the system between each two consecutive evaluation points.

B. UCI repository datasets

Besides the SIGN dataset, we evaluate the presented neuro-fuzzy model on some benchmark problems form the UCI machine learning repository [**?**]. We have focused on problems with more than two classes. The learning process is done in an online mode.

- **Letter** : The objective is to recognize the input vector as one of the 26 English letters. Each letter is represented by a vector of 16 attributes extracted from raster scan image of the letter. The dataset contains 20000 instances. We use 1800 ones in the incremental learning process and the rest is used as evaluation dataset.
- **CoverType** : The aim of this problem is to predict forest cover type from 12 cartographical variables. Seven classes of forest cover types are considered in this dataset. We use in our



(a) SIGN (b) Letters (c) CoverType

(d) Vehicle (e) PenDigits (f) Segment

Figure 2: Performance evaluation during the incremental learning process (the horizontal axis represents the number of learning samples per class, and the vertical axis represents the misclassification rate)

experiment a subset of 2100 instance (1100 train, 1000 test).

- **Vehicle** : The aim is to classify a given silhouette as one of four types of vehicle, using a set of 18 features extracted from the silhouette. The dataset contains 846 instances. We use 640 instances in the learning process, and the rest as evaluation data.
- **PenDigits** : The objective is to classify the ten digits represented by their handwriting information from pressure sensitive tablet PC. Each digit is represented by 16 features. The dataset contains about 11000 instances. 2000 instances are used as learning data, and the rest as evaluation data.
- **Segment** : Each instance in the dataset represents a 3x3 region from 7 outdoor images. The aim is to find the image from which the region was taken. Each region is characterized by 19 numerical attributes. There are 2310 instances in the dataset. We use 1000 instances for the learning and 1300 for the test.

We can note from Figure **??** that our enhanced neuro-fuzzy model outperforms the classic one. Thanks to the improved premise structure, the incremental learning process goes faster and the misclassification rate generally decreases between 10% and 35% depending on the classification problem.

	SIGN	Let.	Cov.	Veh.	Dig.	Seg.
Model I	6.1	23.0	21.0	20.4	3.0	9.0
Model II	4.1	16.0	19.7	16.3	2.0	7.3
MLP	7.4	21.2	17.4	18.6	6.1	4.6
K-NN	4.6	20.9	18.5	28.7	2.3	8.0

Table I: Misclassification rates (%)

We resume in Table **??** the results achieved by the proposed model at the end of the incremental learning process, compared to the classic one and also to two different batch classifiers (Multi-Layer Perceptron (MLP, with one hidden layer) and a K-Nearest Neighbor (K-NN, with K=5)). We note from the table that the presented evolving neuro-fuzzy classifier with a one-pass learning algorithm is even able to outperform in many cases some well-known batch classifiers.

V. Conclusion

An improvement on the premise structure of the first-order Takagi-Sugeno neuro-fuzzy classifier had been presented in this paper. The rotated fuzzy zones of influence in the presented model enable the premise “layer” to consider the correlations that may exist between input features and boost the

overall performance of the classifier. An appropriate recursive premise adaptation method is coupled with a known incremental clustering technique and used with the recursive least squares method to learn the classifier in one-pass incremental mode. The proposed enhancement reduces the misclassification rate by 30% for our specific classification problem (handwritten gesture recognition), and these good results are also obtained on several benchmark classification problems.